FORMAT

A Magazine from INDUG
For DISCiPLE & PLUS D Users

ISSUE #7 - FEBRUARY 1988

THE MAG WITH MORE BYTE...

INDUG.

CONTENTS

ISSUE #7 - FEBRUARY 1988

NEXT MONTH INSIDE THE PLUS D

THE EDITOR'S PAGE

I'm sorry the January issue was a little late going out, but we had a change of printers at the last moment, I hope you found it worth waiting for.

This month sees the end of the series of articles on the Incremental Backup Utility from Nev Young, it's been the biggest program we've printed but I know from your letters and phone calls that it has been well received. Issue #7 sees the start of two new and interesting series. The first, EXPANDING GENS from Dave Kennedy will interest most machine code programmers, there's lots to learn even if you don't use GENS yourself. The other comes from Steve Nutting with a DATA COMPILER program as a preliminary to an irregular series on new features for the GDOS operating system.

We also start a new competition this issue, with over THREE HUNDRED pounds worth of prizes for the winners, not bad I think you will agree. On the subject of competitions I must say how delighted I was with the response to the XMAS WORDSQUARE COMPETITION, so far there's been over sixty entries, the winners will be announced in the next issue. Let me know if you want us to make this a regular feature.

Some months ago I promised the start of a letters page in FORMAT. Well so far there's been no letters to print, apart that is from the letters of praise for FORMAT which I am of course to modest to print (I frame them instead so keep them coming). Lets have a few letters, lets get a few debates going in these pages, lets air a few views. I'm also still looking for writers. Any article on any subject from any angle. Don't be afraid, send it in.

Anyone out there read CRASH? well the Feb'88 issue contains a very good review of the PLUS D by Simon Goodwin, it contains some nice comments about INDUG and FORMAT, good taste that man.

See you next month.

Bob Brenchley. Editor.

DISCiPLE NEWS.

KEMPSTON MOUSE

DISCiPLE / PLUS D owners have finally got a mouse to use with their machines. Kempston Data Ltd have now launched a fully compatible mouse and interface together with a cassette of Toolkit software.

' The Mouse is their usual high resolution two button unit connected to a small interface (same size as their joystick interface) which plugs into the DISCiPLE's rear port. The PLUS D requires the MicroSlot two-way adaptor from MGT. The Toolkit software is a set of machine code routines designed for use from Basic, they give a WIMP environment (Windows, Icons, Menus, Pointers) with lots of extras. A full review is planned for the next issue.

The new Mouse costs £49.95 incl VAT and p & p. Existing Kempston Mouse owners are not left out in the cold either, an upgrade service is available, simply return your unit to Kempston with a cheque for £15 and they will upgrade it and return it with the new software.

Kempston Data Ltd, 22 Linford Forum, Rockingham Drive, Linford Wood, Milton Keynes, MK14 6LY. Tel 0908 690018

SINCLAIR PLANS BIG-BROTHER TO Z88

Uncle Clive may launch a desktop computer as an big brother to his successful Z88 laptop. With the growing sales of the Z88 his new computer will help to establish the business market he failed to reach with the QL. Meanwhile the USA will be seeing its first shipments of the Z88 in the next 6 to 10 weeks.

There is, ALAS, no plans for the launch of a games machine (or so he says).

+3 IN +2

Rumours are floating around about a Spectrum+2.5? This beast, which has been sighted abroad, is outwardly a +2 but with a modified +3 circuit board inside. It is believed that the +3 board is cheaper to make and this is the main reason for the experiment.

Lets hope Amstrad rethink it's launch of the new version of the +2, with the missing lines on the edge connector and it's changed paging system there will be little hardware and software to use with it.

EXPANDING GENS

By: DAVE KENNEDY

GENS, part of the excellent DEVPAC system from HISOFT, has long set the standard for Spectrum assemblers. Few M/C programmers will not have a copy in their collection of utilities. While the program will work with the DISCiPLE / PLUS D you really don't get the best out of the disc system.

Over the next few months I will list a set of routines that will modify version 3.2M of GENS to work more fully with the DISCiPLE / PLUS D disc system and then add a few new commands which I think you will find useful. The modification method is fully described so that it could be altered for another version of GENS.

The routines come in two sections, the first simply alters the GET, PUT and OUT commands to work with disc and adds a CAT command without using any extra memory. The second section adds a 'block line copy' command and a more legible printer listing and uses 420 bytes of memory below GENS. The assembler program still remains totally relocatable.

You should load your version of GENS3 into memory at 27000 and also at high memory, say 40000. The assembler text should be entered in the high memory version and run. The low memory modified version should then be saved before calling so that the relocation code is not altered. All addresses quoted assume that the version of GENS to be modified is loaded at 27000.

When arguments are used with an assembler command, for example P1,999,FILENAME then the binary values of the line numbers will be stored at 34298 #85FA (NUM1) and 34300 #85FC (NUM2) and any other input in the internal buffer at 34256 #85D0. The main SAVE routine is from listing line 110 (OUT1), if the input filename is not already used then the required code block is saved. Otherwise an ERASE option is given and then either a SAVE or exit made. I save the textfile as a code block for speed - it is several times faster than saving as a serial file, although the "T" command is not really usable with a block save.

The CAT command will return to Basic at statement 4 of line 1 this can, of course, be changed. At line 1 I have :-

RAND USR 26580:PRINT USR 60000:GOTO 1: CAT PEEK 16384;"???????
??+":PAUSE 0:GOTO 1

Notice how the tenth character is a "+", I add this to the filename in the GET and PUT routine to produce an easier to read CAT listing. For single drive systems you could alter the listing to default to drive one.

The load command (GET at line 1150) forms the GDOS 'User File Information Area' within GENS modifying the filename buffer as necessary. If the file is not found in the disc directory then a return to GENS input mode is made at line 1240, otherwise the code length details are taken from the header and after file loading the GENS textend marker at 27054 is modified. If the textstart at 34316 does not equal the previous textend then the textfile is renumbered with a step interval of one as a textfile was already present.

The SAVEA routine from line 380 modifies the internal buffer and forms the ten character filename, with the tenth character changed to a "+" so that a CAT listing of assembler only files may be made. A table of the assembler command addresses is stored from 34132 #8554, in the form of letter then call address, this is modified as required by lines 1660.

As my keyboard has an unshifted "-" character, I prefer to use it in place of the ":" as the tape/disc decider in the filename. This modifies the GENS "CP ":" instruction, any other value can be used.

```
10    DK       EQU 27000
10    DK       EQU 27000
20    DK1      EQU DK-1
30    DK2      EQU DK1-1
40
50             ORG #71F6       ;"p" - save textfile
60    PUT      CALL SAVEA-DK   ;"p" = save textfile
70             LD (IX+14),"+"  ;"+" = file identifier
80             JR OUT1
100   OUT      CALL SAVEA-DK   ;"o" = save object code
110   OUT1     PUSH HL         ;code length
120            PUSH DE         ;code start
130   OUT2     RST 8           ;open file header
140            DEFB #35
150            JR NC,OUT4      ;if carry, filename already used
160   R3       CALL ERASE-DK   ;print, beep and input
170            AND 223         ;to upper case
180            CP "Y"
190            JR Z,OUT3       ;if "y" then erase file
200            POP HL
210            POP HL          ;else reset stack and then
220            RET             ;abandon erase & return to input
230   OUT3     LD IX,BUFFER    ;reset "ix" as altered by above
240            PUSH IX
250            RST 8
260            DEFB #41        ;erase file
270            POP IX          ;restore "ix"
280            JR OUT2         ;and repeat open file
290   OUT4     POP DE          ;code start
300            POP BC          ;code length
310            RST 8
320            DEFB #37        ;save code block to disk
330            RST 8
340            DEFB #38        ;close file
350            RET
360
370            ORG #7132       ;overwrite m/drive code
380   SAVEA    LD (BUFFER+16),HL ;code length
390   R6       LD (BUFFER+18),DE ;code start
400   SAVEB    PUSH HL
410            PUSH DE
420   R7       LD HL,BUFFER
430            PUSH HL
440   R8       LD A,(BUFFER+3) ;drive number
450            SUB 48
460            CP 1
470            JR Z,SAVE1      ;if not drive "1" then
480            LD A,2          ;default to drive "2"
490   SAVE1    LD (HL),A
500            PUSH HL
510            RST 8           ;use m/drive select drive hook code
520            DEFB #21        ;to overcome "dos" problem
530            EI              ;disabled by hook code
540            POP HL
550            INC HL          ;recover buffer position
560            LD BC,#0A00     ;b = filename length
570            LD (HL),C
580            INC HL
590            LD (HL),C
600            INC HL
610            LD (HL),"d"
620            INC HL
630            LD (HL),4       ;directory type
640            LD A,13
650   SAVE2    INC HL
660            CP (HL)         ;if n/line then at filename end
670            JR Z,SAVE4
680            DJNZ SAVE2      ;for 10 characters
690            INC HL
700   SAVE4    XOR A
710            CP B            ;if not 10 character length
720            JR Z,SAVE6
730   SAVE5    LD (HL)," "     ;then pad out with spaces
740            INC HL
750            DJNZ SAVE5
760   SAVE6    LD (HL),3       ;code type file
770            LD B,5
780   SAVE7    INC HL          ;bypass code length & start details
790            DJNZ SAVE7
800            LD (HL),C
810            INC HL
820            LD (HL),C
830            INC HL
840            DEC C
850            LD (HL),C
860            INC HL
870            LD (HL),C       ;finish setting up buffer
880            POP IX          ;buffer start
890            POP DE          ;code start
900            POP HL          ;code length
910            RET
```

Next month, the end of the PUT, GET & OUT coding and the start of the new features.

8

BASIC TO TASWORD 2.

By: J.D.RELL

I often need to print out part of a Basic program within an article. At first I printed the top part of the page from Tasword 2, then LLISTed my program, and finally printed the rest of the article. This was OK if the program was to form one block, but just about impossible if the lines were to be interlaced with lines on the page.

If I had resorted to typing in the program lines then errors could have been introduced and these would have been difficult to find. So I needed another method. Now Tasword 2 stores it's files as CODE starting at address 32000. If the Spectrum LLIST could be redirected into memory then my problems would be solved.

This program was originally written for Microdrives and owes a lot to the excellent book "MASTER YOUR ZX MICRODRIVE" by Andy Pennell.

```
10 REM BASIC TO TASWORD 2 FILE CONVERTER.
11 REM V1.3 JUNE 1987.
12 REM WRITTEN BY J.D.R.
20 CLEAR 31999:REM clear space for Tasword file
30 GOSUB 200
50 CLS
60 PRINT INVERSE 1;"BASIC TO TASWORD 2 FILE CONVERTER."
70 PRINT '"Converter loaded.  Now:-"
80 PRINT '"A: LOAD your basic program."
90 PRINT "B: PRESS IN the DISCiPLE's INHIBIT button."
100 PRINT "C: TYPE  'OUT 31,0'"
110 PRINT "D: TYPE  'RANDOMIZE USR 23360'"
120 PRINT "E: TYPE  'LLIST'"
130 PRINT "F: RELEASE INHIBIT button."
140 PRINT '"To SAVE your file type:-"
150 PRINT "SAVE d1""NAME""CODE 32000,PEEK 23353+256*PEEK23354"
160 STOP
200 RESTORE
210 FOR I=23296 TO 23372: READ N: POKE I,N: NEXT I
220 RETURN
300 DATA 254,13,32,15,237,91,57,91,62,63
310 DATA 163,200,62,32,205,28,91,24,245,254
320 DATA 165,56,5,214,165,195,16,12,237,91
330 DATA 57,91,42,178,92,35,25,119,19,237
340 DATA 83,57,91,254,32,40,5,253,203,1
350 DATA 134,201,253,203,1,198,201,0,0,0,0,0,0,0
370 DATA 42,79,92,62,15,141,111,54,0,35
380 DATA 54,91,201
9999 CLEAR : SAVE d1"BASIC-TAS" LINE 10
```

9

TLW-PLUS D

Since Trojan Products took over marketing 'THE LAST WORD' from Myrmidon Software last year, many DISCiPLE users have been putting this excellent word processor to good use.

Alas, new PLUS D owners have not been so lucky, slight hardware differences and the omission of an interrupt vector table in TLW have meant that the program crashed as soon as it was loaded. Well now the problems are solved. Nick Buckenham, the author of TLW (and owner of Myrmidon), has kindly sent in the following alterations to allow TLW to work with the PLUS D.

I must thank Nick for the speed in which he has sorted things out, if only all software companies were as good as this.

1. Without the PLUS D attached to your Spectrum, load TLW from tape as instructed in the manual.

2. Enter as direct commands POKE 65252,190 POKE 65253,191

3. Type CLEAR 40000 then RUN the program. Go into the word processor (Option 1 from menu) then press the ENTER key followed by Extended-Mode 'B' to return to the menu. Now press 'O' to exit to Basic.

4. Repeat step 3 but use CLEAR 30000 instead. This has now cleared 395 bytes at the bottom of the TLW machine code for the new material, given below, to occupy.

5. Now enter these pokes. POKE 64139,192 POKE 49087,195
                            POKE 49088,161 POKE 49089,244

6. As a direct command enter this line of Basic:-
        FOR N=49152 TO 49409: POKE N,191: NEXT N

7. Alter the code file start and length values in line 60 to 49087,16449 (was 49480,16055).

8. Now SAVE the program to a blank tape by doing a RUN 50

9. With your PLUS D attached, 'BOOT' your disc system and load the amended TLW from tape. Then carry out the instructions (in TLW's manual) to convert the program for the DISCiPLE. The mods in step 7 should also now be done to line 80.

10. Save 'THE LAST WORD' to disc by entering RUN 70

Thats all there is to it, if things don't work then run through again and make sure all the POKEs are entered correctly, and as direct commands.

COMPETITION

A new competition, open to all INDUG members, with really FANTASTIC prizes.

Yes our SPRING COMPETITION is a must for everyone. We have lots of prizes including a super NEW PRINTER, Lots of SOFTWARE, stacks of DISCs and much much more. In fact over THREE HUNDRED POUNDS worth of prizes for members to win.

So what have you got to do to win a prize? well it's simple really (well not too difficult) just write a piece of software. It could be a UTILITY, a BUSINESS program, an EDUCATIONAL program or a GAME. It can be in Basic, Machine Code or any other language the Spectrum will understand and provided we can run it.

Each program we receive, and you can submit more than one, will be evaluated by a small team of reviewers and prizes will be awarded on the basis of:-

a) Use of machine.

b) Standard of programming.

c) it's appeal to Spectrum and/or DISCiPLE / PLUS D users.

d) Originality.

Your program need not use the DISCiPLE or PLUS D, it could be written for an unexpanded Spectrum and it wouldn't loose marks. Entries should be on 5.25" 40/80 track, 3.5" 80 track discs or on tape. Please include detailed instructions and a short write-up on how (and why) you wrote the program.

Address to INDUG, 34 Bourton Road, Gloucester, GL4 0LE. and mark the envelope :- SPRING SOFTWARE COMPETITION.

Closing date is 30th April 1988. Winners will be notified by post and the results will be published in the July issue. So get writing, there's no limit to the number of entries you can make. As usual in these things, the Editor's decision is final (and no bribes under £500 will be considered).

The winners, and in fact all other submissions, will also be considered for publication in FORMAT, or on tape, so even if you don't win a top prize you can still earn money from you efforts. A copy of the full rules will be available on request.

GOOD LUCK

BARGAIN CORNER

. _ * _ * _ *

BACK ISSUES BACK ISSUES BACK ISSUES BACK ISSUES BACK ISSUES
 BACK ISSUES BACK ISSUES BACK ISSUES BACK ISSUES BACK ISSUES
  BACK ISSUES BACK ISSUES BACK ISSUES BACK ISSUES BACK ISSUE
S BACK ISSUES BACK ISSUES BACK ISSUES BACK ISSUES BACK ISSU

For members who have missed past issues of FORMAT (or perhaps worn theirs out through constant use) we have now introduced, by popular demand as they say, a back-issue service.

The cost is 60p per copy (80p overseas). Your copies will be sent out as soon as possible but, in order to keep printing costs down, it may take up to 21 days for us to dispatch.

Available Issues

INCREMENTAL BACKUP UTILITY

By: NEV YOUNG

In this, the final article on I.B.U., you will find all the machine code routines you need.

```
; it has to start somewhere so lets start at 65000 leaving 535
  bytes for code
;
ORG 65000
THE_START:
;
; A routine  for filling any area of memory with a single byte.
  The start address is held in the word var 'clr_start' & the
  end  address is held in the word var 'clr_end'. The byte to be
  placed into the area is held in 'clr_byte'. It will be placed
  into all memory locations inclusive of  the start and end
  addresses.
;
CLR_START: DEFW 00
CLR_END: DEFW 00
CLR_BYTE: DEFB 0
;
CLR_MEM:
  LD HL,(CLR_END)
  LD DE,(CLR_START)
  SBC HL,DE
  LD B,H
  LD C,L ; BC now contains the number of bytes to fill
;
  LD HL,(CLR_START)
  LD D,H
  LD E,L
  INC DE ; DE points to (CLR_START+1)
  LD A,(CLR_BYTE) ; get the byte
  LD (HL),A ; and put it into the first location
  LDIR ;  now clear the memory
  RET ; and return
;
; a routine to compare two areas of memory. the start  addresses
  are held in he word variables 'BASE_1 and BASE_2' and the
  number of bytes to compare is held in the single byte variable
  'C_LEN'
;
; the routine will return with BC = 0 if the compare is ok
  otherwise it will return with B = to the number of bytes to
  compare and C = to the value of the difference (XORed) of the
  two bytes that caused the compare failure
;
BASE_1: DEFW 00
BASE_2: DEFW 00
```

```
C_LEN: DEFB 0 ; compare 256 bytes
  DEFB 0 ; used to ease programming
;
COMP_1: ; start here
  PUSH IX ; save IX
  LD HL,(BASE_1)
  LD IX,(BASE_2)
  LD BC,C_LEN
;
COMP_LOOP: ; main loop
  LD A,(HL)
  XOR (IX)
  JR NZ,COMP_END
  INC HL
  INC IX
  DJNZ COMP_LOOP
;
COMP_END: ; the exit route
  LD C,A ; the difference
  POP IX ; restore IX
  RET ; go home
;
;
; This routine will build a sector map in memory for the
  disciple discs. Each bit is one sector and corresponds to the
  bit map contained within the file headers. The bits are chosen
  from a 256 byte area of memory that is assumed to be a
  disciple disc header. The start address is contained in the
  word variable map_addr and the area to build the map starts at
  the address contained in the word variable 'MAP_STORE'. The
  code itself is relocatable
;
;
MAP_ADDR: DEFW 00
MAP_STORE: DEFW 00
;
MAP_BUILD:
  PUSH IX ; save IX
  LD HL,(MAP_ADDR)
  LD IX,(MAP_STORE)
  LD B,0 ; map all 256 bytes
;
; I am well aware that the map only goes from byte 14 of the
  sector through to byte 210. If you are concerned about a few
  millisecs then change it.
;
MAP_LOOP:
  LD A,(IX)
  OR (HL) ; logical or of the map bits
  LD (IX),A
  INC IX ; move on to the next byte
  INC HL
  DJNZ MAP_LOOP
;
;  fall through to the return routine
;
  POP IX
  RET
```

```
;
; Here we have a routine that, given a disciple sector map, will
  return with bc = the sector number of the next sector used
  within the map. In this case the first sector on the disc is
  0. The lowest sector returned is 40 ie the first sector after
  the directory
;
; The track and sector can be calculated from
      t = INT (BC / 10)
      s = 1 + BC - t * 10
  all that then remains is to correct for the discontinuity when
  you change sides. The routine is very repetitive but it works.
  Also you must note that as part of its function it will
  destroy the sector map. When there are no more bits set in the
  map it will return with BC = 0FFFFH.
;
; It is intended that this routine be used with the map_build
  routine and uses the same map_store
;
NEXT_START:
  LD BC,39
  LD DE,15
  LD HL,(MAP_STORE)
  ADD HL,DE
  LD DE,210-15
;
NEXT_0: ; test the first bit in this byte
  INC BC ; inc the sector number
  BIT 0,(HL) ; do the test
  JR Z,NEXT_1 ; if not set move on to the next bit
  RES 0,(HL) ; otherwise reset it
  RET ; and return
;
NEXT_1: ; same as before but for a different bit
  INC BC
  BIT 1,(HL)
  JR Z,NEXT_2
  RES 1,(HL)
  RET
;
NEXT_2:
  INC BC
  BIT 2,(HL)
  JR Z,NEXT_3
  RES 2,(HL)
  RET
;
NEXT_3:
  INC BC
  BIT 3,(HL)
  JR Z,NEXT_4
  RES 3,(HL)
  RET
;
NEXT_4:
  INC BC
  BIT 4,(HL)
  JR Z,NEXT_5
```

```
   RES 4,(HL)
   RET
;
NEXT_5:
   INC BC
   BIT 5,(HL)
   JR Z,NEXT_6
   RES 5,(HL)
   RET
;
NEXT_6:
   INC BC
   BIT 6,(HL)
   JR Z,NEXT_7
   RES 6,(HL)
   RET
;
NEXT_7:
   INC BC
   BIT 7,(HL)
   JR Z,NEXT_8
   RES 7,(HL)
   RET
;
NEXT_8:  ; done all the bits in this byte so lets move on to the
next
;
   INC HL
   DEC DE
   LD A,D ; test if all bytes done
   OR E ; DE = 0 if all done
   JR NZ,NEXT_0 ; more to do so jump back
   LD BC,0FFFFH ; report end
   RET        ; and go home
;
THE_END: ; mark the end address
;
; this is the linkage section. here all the memory locations
  used are listed so that they can be peeked at by basic basic
  should then store the addresses in similar named variable and
  use that to reference the machine code variables
;
ORG 65500 ; start at a known address.
DEFW THE_START
DEFW CLR_START
DEFW CLR_END
DEFW CLR_MEM
DEFW BASE_1
DEFW BASE_2
DEFW COMP_1
DEFW MAP_ADDR
DEFW MAP_STORE
DEFW MAP_BUILD
DEFW NEXT_START
```

Well that's it, I hope I have explained things well enough and I hope you find I.B.U. as valuable as I do. For those of you who dread the task of typing in the program turn to the USE page.

DATA COMPILER

By: STEVE NUTTING

This data compiler is going to be a very important machine code routine over the coming FORMAT issues. The routine compiles special REM data values at high speed in less than a second and then dumps the code to Spectrum memory. The routine is fully error trapped and will display helpful error messages. Type in the long listing and RUN it if all o.k. the program will automatically save the machine code routine to disc, keep this in a safe place you will need it soon.

Lets try an example listing which will help you to understand what the compiler does and how to correct errors. Type this in:-

```
 5 CLEAR 29999: RANDOMIZE USR 64512
10 REM s30000
20 REM 62,70,33,0,64,17,0,24,117,84,30,1,237,176
30 REM 119,1,255,2,237,176,201
40 REM e
```

Then type CLEAR 64511: LOAD d1 "datacomp" CODE, Type RUN and you will see on top of the screen The Compile address which is where the following bytes will be poked into memory, note line 10 REM s defines the address. You may just see on the screen numbers after the LINE wizz past, this shows which line the compiler in working on. It should end up showing line 40. Note REM e tells the compiler when to stop. This should always be at the end of a listing.

Lets now make some alterations to the listing to force error messages, first alter line 10 to become REM s64512 now type RUN, you should now come up with the error message ADDRESS OUT OF RANGE BETWEEN RAMTOP+1 TO 64511. This means you have tried to compile bytes below RAMTOP or above 64511 which may crash the computer. Now alter line 10 to REM s40000 and alter byte 70 in line 20 to 700 then type RUN. This time the error message BYTE OUT OF RANGE BETWEEN 0-255 20:2, this means that in line 20 and the 2nd byte in that line is not in range between 0-255. So press EDIT and alter byte 700 to 70, note no need to LIST 20 then EDIT, my routine automatically sorts the line so you only need to press EDIT. Next alter line 30 to become 30 REM 19,1,255,237,176,201:991 and type RUN. You now should see CHECKSUM DATA ERROR 30:7, meaning the the bytes in line 30 do not add up to 991 because you either entered a wrong byte (in this case 19 should be 119) or you had too many or too few data statements in line 30 (in this case we missed out data byte 2). If we now alter line 30 to become REM 119,1,255,2,237,176,201:991 and type RUN we shouldn't get an error message. Now delete line 10 and type RUN, you should get the error message NO COMPILE ADDRESS AT START. This means you have not entered the REM s address to define the memory location where the following bytes will be poked into.

The first special REM listing will appear in the next issue of format with others to hopefully follow soon after.

```
  1 CLEAR 64511: LET c=0: FOR a=64512 TO 65392: READ n: POKE a,n
  : LET c=c+n: NEXT a: IF c<> 96318 THEN PRINT "DATA ERROR": STOP
  5 SAVE d1 "datacom p" CODE 64512,881: STOP
 10 DATA 62,2,205,1,22,33,152,254,205,140,254,42,83,92,35,35,94,
35,86,25,17,6,0,25,126,203,175,254,83,194,10,253,35,205,223,253,
58,132,254,254,13,194,5,253,34,128,254,205
 20 DATA 29,253,34,136,254,62,22,215,175,215,62,16,215,237,75,12
8,254,205,43,45,205,227,45,42,126,254,70,35,78,35,35,35,35,229,2
37,67,73,92,175,50,133,254,62,22,215,175
 30 DATA 215,62,28,215,205,27,26,225,126,254,101,200,254,112,202
,57,253,254,80,202,57,253,254,69,200,254,115,202,20,253,254,83,2
02,20,253,34,126,254,42,126,254,205,223
 40 DATA 253,124,167,194,15,253,58,133,254,60,50,133,254,42,128,
254,205,29,253,58,124,254,119,35,34,128,254,58,132,254,254,44,40
,217,254,58,32,152,42,126,254,205,223,253
 50 DATA 229,33,0,0,34,130,254,42,128,254,58,133,254,95,22,0,167
,237,82,67,94,22,0,229,42,130,254,25,34,130,254,225,35,16,241,20
9,42,130,254,167,237,82,202,71,252,33,36
 60 DATA 255,205,140,254,62,13,215,237,75,73,92,205,43,45,205,22
7,45,62,58,215,58,133,254,60,79,6,0,205,27,26,207,8,225,33,3,255
,24,219,33,229,254,24,214,33,58,255,24
 70 DATA 209,35,195,33,252,33,92,255,24,200,84,93,124,254,252,40
,12,237,75,178,92,167,43,237,66,56,2,235,201,225,33,180,254,205,
140,254,207,8,126,35,254,13,32,250,70,35
 80 DATA 78,237,67,73,92,35,35,35,235,33,212,48,34,134,254,235,1
75,50,138,254,50,139,254,126,254,234,192,35,126,254,101,200,254,
69,200,254,115,202,20,253,254,83,202,20,253
 90 DATA 34,126,254,42,126,254,205,223,253,68,77,237,91,138,254,
25,34,138,254,96,105,237,91,134,254,25,34,134,254,17,212,48,167,
237,82,237,91,136,254,25,229,94,35,86
100 DATA 235,17,179,26,25,235,225,115,35,114,42,134,254,17,46,52
,167,237,82,210,24,253,33,133,254,52,58,132,254,254,44,40,182,25
4,58,32,16,42,126,254,205,223,253,237,91
110 DATA 138,254,167,237,82,194,226,252,42,126,254,70,35,78,35,3
5,35,237,67,73,92,195,81,253,235,33,119,254,175,6,5,119,35,16,25
2,50,124,254,50,125,254,235,6,0,126,254,58
120 DATA 48,8,254,48,56,4,4,35,24,243,50,132,254,35,34,126,254,1
20,254,6,210,4,253,43,43,17,123,254,126,214,48,18,43,27,16,248,2
21,33,119,254,221,126,0,167,40,13,71,42
130 DATA 124,254,17,16,39,25,16,253,34,124,254,221,126,1,167,40,
13,71,42,124,254,17,232,3,25,16,253,34,124,254,221,126,2,167,40,
13,71,42,124,254,17,100,0,25,16,253,34,124
140 DATA 254,221,126,3,167,40,13,71,42,124,254,17,10,0,25,16,253
,34,124,254,221,94,4,22,0,42,124,254,25,34,124,254,201,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,126,215
150 DATA 35,203,126,40,249,126,230,127,215,201,67,79,77,80,73,76
,69,32,65,68,68,82,69,83,83,58,32,32,32,32,32,32,76,73,78,69,
186,13,18,1,65,68,68,82,69,83,83,32,79
160 DATA 85,84,32,79,70,32,82,65,78,71,69,32,66,69,84,87,69,69,7
8,13,82,65,77,84,79,80,43,49,32,84,79,32,54,52,53,49,177,13,18,1
,78,79,32,67,79,77,80,73,76,69,32,65,68
170 DATA 68,82,69,83,83,32,65,84,32,83,84,65,82,212,13,18,1,85,8
3,69,68,32,54,32,68,73,71,73,84,83,32,79,82,32,77,79,82,69,32,79
,86,69,82,70,76,79,215,13,18,1,67,72,69,67
180 DATA 75,83,85,77,32,68,65,84,65,32,69,82,82,79,210,13,18,1,6
6,89,84,69,32,79,85,84,32,79,70,32,82,65,78,71,69,32,66,69,84,87
,69,69,78,32,48,45,50,53,181,13,18,1,80
190 DATA 76,85,83,32,68,32,79,70,83,69,84,32,69,82,82,79,210,120
```
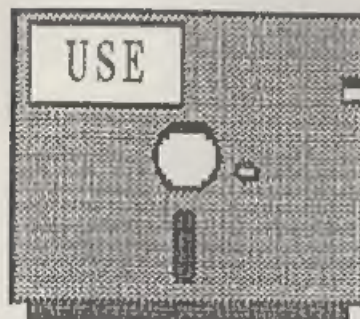

USERS SOFTWARE EXCHANGE

The User Software Exchange (USE for short) provides DISCIPLE and PLUS D owners with a growing range of software specially designed for their systems.

Here is the latest list of software for sale to INDUG members. The software is supplied on tape, for easy transfer to any format/size of disc, thus keeping costs as low as possible.

Tape No	Title	Program Description
T001	LCOPY	Routines to replace the GDOS printer dump routines in your system file. Enables Epson compatible printers without Esc * to be used with both SCREEN$ 1 & 2 and SNAPSHOT prints.
T002	MULTI-POKE	Program to display and edit 48K SNAPSHOT files. Full facilities for entering published POKEs i.e. Infinite Lives etc. Makes use of printer if attached.
T003	G-HACKER	A graphic investigator. Look inside commercial programs, find the Sprites and pictures. Works in 48K mode but will handle many 128K programs.
T004	I.B.U.	The highly successful 'Incremental Backup Utility', written by Nev Young.

Each tape, costs £3.95, or £4.95 including airmail postage for overseas members. Instructions are supplied where needed and all tapes are professionally recorded. Send your order (on a separate piece of paper please), clearly stating the Tape Number; Title; Quantity required and your membership number, to:- INDUG, USE ORDERS, 34 Bourton Road, Gloucester, GL4 0LE, England. Cheques (drawn on a UK bank), Postal Orders, Euro Cheques or Cash accepted. Payment must be in STERLING. Please allow 28 days for delivery.

P.S. We urgently need more software, send your latest creation for evaluation, it may be worth more than you think.